Secret scanning: a key to your cybersecurity strategy

WRITTEN BY GITHUB WITH

What's inside

Introduction

In 2024, over 39 million secrets were detected on GitHub. As developers leverage agentic assistants like GitHub Copilot to write code more quickly than ever before, developers are also leaking secrets at unprecedented rates.[1] This combination of expanded attack surface, an explosive volume of leaked secrets, and increased potential for human error means that malicious actors now have even more avenues to gain unauthorized access to cause devastating damage.

Unfortunately, bad actors constantly scan platforms for exposed credentials, and a leaked secret can be discovered and exploited within minutes or even seconds. This is why securing access to systems and sensitive information is more important than ever.

Companies need a new strategy for protecting their secrets from compromising leaks. In this ebook, we'll outline what you need to know about secret detection and protection, highlight why it should be a core part of your application security program, and how to get started securing your secrets today.

1: GitHub. "The Next Evolution of GitHub Advanced Security." GitHub Blog. February 23, 2024. https://github.blog/security/application-security/next-evolution-github-advanced-security/.

What are secrets?

Secrets are the sensitive information used to authenticate, authorize, or encrypt data within an application or system. These digital keys or credentials are crucial for secure operations but can pose significant risks if exposed. Secrets include API keys, passwords, tokens, encryption keys, SSH keys, and other sensitive information granting access to various resources or services.

Secrets are essential for modern development workflows, enabling applications to securely communicate with databases, third-party APIs, cloud services, and other critical infrastructure. They can include:

- A web application using an **API key** to authenticate requests to a payment gateway
- A **service provider access key** hard-coded in a cloud deployment script
- A database connection string with login **credentials** in an application's configuration file

While these secrets are vital for functionality, they should never be exposed in any location.

Secret protection becomes increasingly complex in distributed development environments where code is shared, reviewed, and deployed across multiple platforms, by development teams and even AI agents. While there are technologies for managing and encrypting keys, deploying them across all development use cases is difficult.

As organizations scale their development efforts and adopt more cloud-based services, the number of secrets in circulation grows, amplifying the potential impact of any leak or breach. This is where robust secret management practices, including secret detection and remediation, become crucial for maintaining security and compliance in the software development lifecycle.

The costs of secret leaks

Secrets are usually the first step for bad actors to launch broader attacks. Leaked credentials and API keys can be used to gain direct access to personal information, financial data, or intellectual property. **In 2025, the average total cost of a data breach was $4.4M.**[2]

Once attackers access one system, they can move to other areas of an organization's network, often undetected. The cost can be immense and multifaceted.

- **Financial penalties:** Breaches from leaked secrets can be subject to regulatory fines and legal fees. For example, in 2019 the FTC fined a large American credit bureau $700M for a data breach affecting 147 million people.[3]
- **Remediation and incident response:** Fixing the problem is not straightforward and can be time-consuming. In larger incidents, a 3rd party may be called in to ensure the breach is contained and resolved.
- **Operational downtime:** Organizations may be forced to pause operations or incur downtime while addressing the issue. This can be especially prominent in healthcare, where leaked secrets have led to ransomware attacks forcing hospitals to cease operations for multiple days or even weeks.
- **Reputational damage:** And then there's what could be the most devastating, but also the most difficult to measure: damage to an organization's reputation. If the customers lose trust, it can take years to win them back, if at all.

Organizations need a strategy for securing their growing secret footprint to lower their cybersecurity risk. That starts with understanding where and when secret leaks may occur, and what attacks may result.

2: IBM Security. "Cost of a Data Breach Report 2025." IBM. 2025. https://www.ibm.com/reports/data-breach.

3: Federal Trade Commission. "Equifax to Pay $575 Million as Part of Settlement with FTC, CFPB, and States Related to 2017 Data Breach." Press release. July 22, 2019. https://www.ftc.gov/news-events/news/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related-2017-data-breach.

How secrets can be exploited

When a secret is exposed, it often serves as a critical entry point for attackers to initiate a more extensive and damaging breach. Here are some of the tactics a malicious actor can leverage once they possess a leaked secret:

- **Initial access:** Attackers leverage exposed secrets to gain legitimate credentials to access a system or service. This initial foothold bypasses traditional security measures since the attacker looks like an authorized user.
- **Reconnaissance:** Once inside, attackers can use the compromised secret to gather information about the target environment. They may access logs, configuration files, and databases, map the network architecture, and identify additional targets.
- **Privilege escalation:** With knowledge of the system, attackers often seek to elevate their privileges. They might exploit vulnerabilities or use the exposed secret to access more sensitive areas of the network, potentially gaining administrative rights.
- **Lateral movement:** Using the elevated privileges, attackers may move laterally across the network, compromising additional systems and services. This expansion increases their reach and the potential impact of the breach.
- **Data theft:** As the attack progresses, sensitive data becomes a prime target. Attackers may use the compromised systems to locate and extract valuable information, such as customer data, financial records, or intellectual property.
- **Persistence:** To maintain long-term access, attackers might create backdoors or compromise additional accounts. This enables them to return even if the initial exposed secret is discovered and revoked.

- **Supply chain attack:** In some cases, attackers may use the compromised environment to target connected third-party vendors or customers, turning a single breach into a far-reaching supply chain attack.
- **Infrastructure-as-Code (IaC) exploitation:** If the exposed secret grants access to cloud resources or CI/CD pipelines, attackers can manipulate IaC configurations, potentially compromising entire cloud environments or development processes.
- **Ransomware:** With broad access to the network, attackers may deploy ransomware, encrypting critical data and systems across the organization, leading to operational disruption and potential financial losses.

The cascading nature of these attacks underscores the critical importance of protecting secrets. A single exposed credential can lead to a comprehensive breach, potentially costing millions in addition to untold damage to an organization's reputation.

> "Data breaches can also expose companies to reputational damage, erosion of trust among customers and customer churn. For chief data officers and other data leaders—or anyone else responsible for data strategy, governance and value creation—this research is a wakeup call."
>
> IBM Cost of a Data Breach[4]

4: IBM Security. "Cost of a Data Breach Report 2025." IBM. 2025. https://www.ibm.com/reports/data-breach.

Three approaches to protecting your secrets

Protecting secrets is a top priority in today's software ecosystem, especially as AI and automation are accelerating the pace of software development. Effective secret protection means meeting developers where they are—using developer-native tools with built-in, automated security. If secret protection creates friction or disrupts their flow, developers may bypass controls or lose trust in their tools, leading to increased risk.

To build a resilient secret protection program, the most imperative thing to do is to set prevention mechanisms in place, like push protection, to stop new secrets from leaking. If a secret is bypassed, you'll still have an audit trail, which will be helpful with incident response. Beyond that, you should take care to implement a secret management program to centralize, standardize, and automate secrets management across your organization, including lifecycle management of secrets, access control, and accounting across these systems.

There are three approaches you could take:

1. **Reactive:** A reactive approach focuses on responding to incidents after they occur. This includes leveraging secret scanners to detect any potentially leaked secrets, as well as implementing well-documented incident response procedures for rapid secret revocation and rotation following a breach, as well as automated forensic tools for analyzing impact and root cause. Post-incident analysis will help you learn from vulnerabilities in order to prevent future issues. A reactive approach alone leaves organizations vulnerable between breach occurrence and detection—especially given today's automated threat actors and "zero-time-to-exploit" risks.
2. **Proactive:** Proactive secret management emphasizes prevention and automation. This includes:
    - **Prevention:** Blocking secrets before they're introduced into your codebase, using solutions like GitHub push protection.

- **Automated secrets management:** Limiting or removing human interaction with automated secrets pipelines (e.g. creation, rotation, etc.), using dynamic secrets, and automatic rotation of static secrets.
- **Zero Trust and RBAC:** Implementing least privilege policies and dynamic access for both human and machine users.

3. **Hybrid:** A hybrid approach layers security by implementing regular lifecycle management, secret rotation, and access control while maintaining reactive incident response protocols for added agility. Beyond an emphasis on preventative measures, continuous monitoring with always-on security scans and auditing are also key components, allowing teams to detect issues in real-time and respond rapidly when needed. The hybrid approach provides a flexible, robust security posture, enabling organizations to proactively prevent issues while remaining prepared to react to unexpected events.

How to create and implement a secret protection strategy

Creating a strong secret protection strategy is essential for reducing cyber risk and advancing your DevSecOps practice. Preventing secret leaks requires both effective tools and a culture of trust and collaboration between security teams and developers. Below are actionable steps to guide your approach:

- **Pinpoint risk areas:** Identify your most vulnerable points for secret leaks. Use tools and processes—including GitHub secret scanning—to locate where secrets reside, such as in code, automation pipelines, cloud services, and with third-party vendors. Engage with stakeholders, especially those with privileged access, to uncover potential blind spots.
- **Educate and raise awareness:** Ensure everyone involved in development understands why secret protection matters. Launch regular, targeted training sessions that highlight real-world incidents and provide practical prevention tactics. Foster a blame-free culture that encourages learning from mistakes, and keep teams updated on evolving best practices. (See Microsoft Learn[5] for more guidance.)
- **Integrate automated controls:** Incorporate secret detection and management directly into developer and automation workflows. Choose solutions that integrate seamlessly with your version control, CI/CD environments, and tools like GitHub issues and pull requests. Prioritize platforms that offer automated scanning, push protection, and rapid incident response.
- **Deploy secret management tools:** Use dedicated services like Azure Key Vault, HashiCorp Vault, or Google Secrets Manager[6] to securely store and manage your most sensitive secrets. Take advantage of features like auto-rotation, role-based access, and auditing.
- **Continuously scan for exposures:** Automated scanning should cover code repositories, pipelines, infrastructure as code, logs, binaries, and other data artifacts. Keep detection ongoing to quickly find and address exposures—even in unexpected places.

5: Microsoft. "Best Practices for Protecting Secrets." Microsoft Learn. Accessed December 22, 2025. https://learn.microsoft.com/en-gb/azure/security/fundamentals/secrets-best-practices.

6: Google Cloud. "Secret Manager." Google Cloud. Accessed December 22, 2025. https://cloud.google.com/security/products/secret-manager?hl=en.

- **Rotate secrets regularly:** Automate policies for rotating secrets throughout their lifecycle, from creation and distribution to revocation and secure destruction. Use short-lived or just-in-time secrets wherever possible.
- **Limit access with least privilege:** Restrict secrets to only those who need them, and consistently apply least privilege principles. Review permissions frequently as teams and projects change.
- **Monitor and audit access:** Track and review logs for unauthorized access attempts. Use context-aware tools to prioritize alerts and streamline incident response.
- **Extend protections to supply chain and partners:** Apply your secret protection standards to SaaS services and partners, monitoring for exposures both inside and outside your organization's direct control.
- **Adopt built-in solutions:** Choose integrated detection features to boost adoption and minimize tool fatigue, helping teams stay productive while staying secure.
- **Align with regulations and standards:** Stay current with regulatory requirements (like DORA, PCI DSS v4, SEC, and GDPR/CCPA) and align practices with frameworks like OWASP, NIST, and the OWASP Secrets Management Cheat Sheet.[7]
- **Embrace continuous improvement:** Treat secret management as ongoing. Regularly reassess your risk landscape, update controls as your tech stack evolves, and respond to new threats and compliance needs. Maintain an incident response plan that includes automated revocation and clear stakeholder communication.

By combining clear risk assessment, education, integrated technology, automated controls, and a learning mindset, your organization can significantly lower the risk of secret leaks and build a future-proof, resilient secret management program.

7: OWASP Foundation. "Secrets Management Cheat Sheet." OWASP Cheat Sheet Series. Accessed December 22, 2025. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html.

How to secure your secrets with GitHub Secret Protection

If we learned one thing from the state of The Octoverse in 2025,[8] it's that the developer ecosystem is growing rapidly. With all the potential for innovation this brings, it also brings along risk, which is why we're responding to that risk by expanding our secret scanning and protection tools.

GitHub Secret Protection continuously monitors your GitHub perimeter, helping you prevent exposures, protect credentials, and ship securely every day. Fortified by a global security partnership of over 150+ providers to ensure the highest levels of detection accuracy, GitHub Secret Protection is a powerful way to mitigate risk, no matter the size of your organization.

In 2024, 39M secret leaks were detected with Secret Protection and 4.4M secrets were prevented from leaking on GitHub.

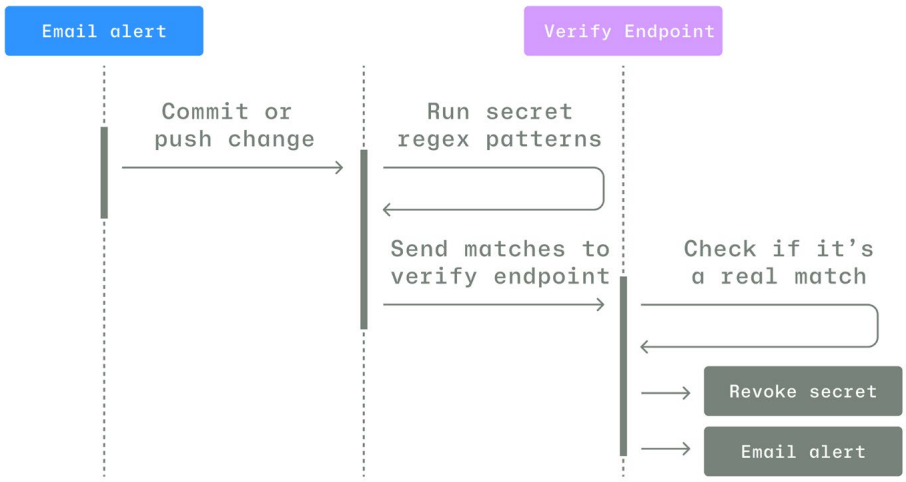
Eight GitHub Secret Protection features to keep your secrets secure from the start

- **Push Protection:** Automatically block commits containing secrets before they reach your repositories, helping prevent leaks at the source. This feature is available for public and private repos, and supports enforcement at enterprise scale.
- **Secret scanning alerts and validity checks:** Get real-time alerts and notifications when secrets are detected. Built-in validity checks show whether detected secrets are still active so you can prioritize remediation.
- **Custom secret patterns:** Define custom patterns to detect proprietary, organization-specific, or project-specific secrets

8: GitHub. "The State of the Octoverse." Accessed December 22, 2025. https://octoverse.github.com/.

in addition to standard credentials.

- **Partner program and automated provider notification:** Notify participating service providers (e.g., AWS, Google, OpenAI) when their tokens are found in public repos—enabling rapid revocation, quarantine, or other mitigation actions.
- **Security overview dashboard:** Access an organizational dashboard to view the distribution of secret risks, trends, and outstanding exposures across all your repositories.
- **Governance, policy enforcement and delegation:** Enforce policies—such as who can dismiss alerts or bypass push protection—and support auditable delegation to ensure compliance and internal governance.
- **Integration and automation:** Automate incident response management via webhooks and APIs for automated monitoring, alerting, and remediation. Security teams can also attach custom remediation guidance to alerts.
- **Security campaigns and compliance support:** Launch and manage remediation campaigns across your organization in order to address existing secret leaks.



For developers, the best way to protect your secrets is to enable these protections, understand your risk distribution within your codebases, and foster a culture where security is always top of mind.

Want to keep your secrets safe and secure?

→ Learn more about GitHub Secret Protection today.